Patent
45098.00014.UTL1

UTILITY APPLICATION

UNDER 37 CFR § 1.53(B)

TITLE: SYSTEMS AND METHODS FOR SYNCHRONIZING DATA BETWEEN COMMUNICATION DEVICES IN A NETWORKED ENVIRONMENT

APPLICANT(S): Dan Jones et al.

Correspondence Enclosed:

Utility Application Transmittal Sheet (1 pg.) Utility Application Cover Sheet (1 pg.); Specification (32 pgs.); Drawings Figures (4 pgs.); Nonpublication Request under 35 U.S.C. 122(b)(2)(B)(i) (1 pg.) and Return Postcard.

"EXPRESS MAIL" Mailing Label Number EL 997091117US
Date of Deposit: December 12, 2003
I hereby certify under 37 CFR §1.10 that this correspondence is being deposited with the United States Postal Service as "Express Mail Post Office to Addressee" with sufficient postage on the date indicated above and is addressed to the Mail Stop Patent Application, Commissioner for Patents, P. O. Box 1450, Alexandria, VA 22313-1450.

Maria Carvalho
Maria Carvalho

SPECIFICATION

SYSTEMS AND METHODS FOR SYNCHRONIZING DATA BETWEEN COMMUNICATION DEVICES IN A NETWORKED ENVIRONMENT

BACKGROUND

1. Field of the Invention

[001] The field of the invention relates generally to network communications and more particularly to synchronizing data shared between a plurality of networked communication devices.

2. Background Information

[002] Networked communication devices often need to share information. For example, the need often arises in networked environments where data on one device needs to be replicated on one or more other devices. It will be understood that even the simple task of sharing data can be problematic in a networked environment; however, problems are further complicated when the data to be shared is dynamic or changing over time in such a way that only the latest set of data is of value. Various approaches exist for accomplishing the replication of dynamic data in a networked environment; however, conventional approaches are limited and often ineffective.

[003] One conventional approach to sharing data in a networked environment uses repositories of data known as buffers on each of the devices. The buffers can then be synchronized by sending messages over a network connection. In order to reduce the amount of data sent and thus the usage of network connection bandwidth, an entire copy of the original, or source buffer, can be sent to one or more destination buffers. Thereafter, only

changes in the source buffer are sent at various intervals. The destination device can then apply the changes to its local buffer to maintain synchronization with the source buffer. This approach allows the update interval to be selected to match the desired, or available network bandwidth between the source and receiving device. Selecting the update interval, however, can be problematic.

[004] For example, updates can be sent to a destination device anytime a change is made to the buffer on the source device. But this can be ineffective because the destination device or interconnecting network link may be incapable of accepting and processing the updates at the rate changes occur at the source device. Accordingly, updates must either be discarded, resulting in loss of synchronization, or queued-up, in which case a lag develops between source and destination devices corresponding to the length of the queue. Such loss of synchronization or lag between destination and source devices can lead to problems. Moreover, queues may also consume significant and potentially unbounded resources, leading to further problems.

[005] Alternatively, updates can be sent when requested by the destination device. This allows updates to be sent at a rate that they can be processed, but the receiving buffer is only synchronized with the source buffer at times when an update is sent. Thus, the source buffer may go through several intermediate states in the interval between updates. These intermediate states will not be reflected in the destination buffer.

[006] A further drawback to conventional approaches can occur when a plurality of destination buffers must be synchronized with a source buffer. Often the data handling capability of each destination differs. Further, the network connections between source and each of the destination devices are not necessarily identical in terms of bandwidth, speed,

latency, or reliability. As a result, changes sent to the destination devices can be sent only as frequently as can be handled by the slowest connected device or network connection. Accordingly, devices capable of receiving more information or more intermediate states are not able to operate using their full capability.

[007] For example, a multimedia collaboration session, where a user's computer display, or region of the display, is shared with remote viewers, can be used to illustrate the problems with conventional data sharing approaches. The portion of the display to be shared is often captured in a buffer and transmitted to other viewers' computers. As the display changes, the source buffer is updated and updates are sent to destination buffers and displayed on viewers' displays; however, these updates are sent only at the rate corresponding to the slowest of all the connected networks and devices. Accordingly, even users with fast computers will experience less frequent updates and unpleasant artifacts such as jerkiness, low frame rate, and poor quality in displays involving changes or motion.

[008] Alternatively, a separate instance of the source buffer can be maintained for each destination device and separate computation of changes. And update message transmission can be performed for each connected destination device. This technique allows each device to receive updates at a rate that best uses available network and device capabilities; however, this approach suffers from a limitation in that maintaining buffers and computing changes requires large amounts of memory and processing power. Thus, maintaining a separate buffer instance for each connected destination limits the number of endpoints that can be simultaneously connected. This is a serious limitation in a system such as a multimedia collaboration system, which may be of use only if a certain number of parties are able to connect.

[009] Thus, a significant implementation challenge exists in synchronizing multiple destination buffers and devices to a source buffer containing data that changes over time. This is especially true when the data handling capacity of connected destination devices are not equal, as is typical in conventional networked environments. If all destinations devices are sent updates for every change in the source buffer, the volume of data may overwhelm the capacity of some devices and network links, resulting in loss of synchronization for those devices. If data is sent at a rate compatible with all devices, i.e. sent at the rate of the slowest receiving device and network link, devices with greater capability will receive poor quality data. If a separate data stream is created for each connected device, the resources of the sending device may become taxed and the result will be a limit to the number of destination devices that can connect simultaneously.

SUMMARY OF THE INVENTION

[010] A source communication device comprises a cluster manager to group destination device connections into a plurality of performance clusters. The cluster manager can then assign a synchronization mechanism to each performance cluster. Each synchronization mechanism can then allow data to be optimally shared between the source device and destination devices within a given performance cluster.

[011] These and other features, aspects, and embodiments of the invention are described below in the section entitled "Detailed Description of the Preferred Embodiments."

BRIEF DESCRIPTION OF THE DRAWINGS

[012] Features, aspects, and embodiments of the inventions are described in conjunction with the attached drawings, in which:

[013] Figure 1 is a diagram illustrating an example network communication system configured in accordance with one embodiment;

[014] Figure 2 is a diagram illustrating another example embodiment of a network communication system configured in accordance with one embodiment;

[015] Figure 3 is a flow chart illustrating an example method for configuring performance clusters within a source device included in the network communication system of figure 1 in accordance with one embodiment; and

[016] Figure 4 is a flow chart illustrating an example method for synchronously sharing data using performance clusters established using the method of figure 3 in accordance with one embodiment.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

[017] Figure 1 is a diagram illustrating an example network communication system 100 configured in accordance with one embodiment of the systems and methods described herein. Network communication system 100 comprises a source device 101 interfaced with a plurality of destination devices 111 via connection interfaces 103a, 104a, and 105a and communication connections 106-110. Source device 101 comprises a source data buffer 102 that comprises data to be replicated on each destination device 111, e.g., in an associated destination data buffer 117. Source device 101 can also include one or more synchronization mechanisms, such as synchronization mechanisms 103, 104, and 105, as well as a cluster manager 118. The source data buffer 102 can be coupled directly or indirectly to synchronization mechanisms 103, 104 and 105.

[018] The number of destination devices 111 and the number of associated communication connections are shown for illustration only. Only a small number of

connections and destination devices 111 are shown for clarity and should in no way imply a limit or suggestion as to the number of communication devices 111 that can be supported using the systems and methods described herein.

[019] Often, communication connections 106-110 can have different performance capabilities. As described below, performance capability can be defined in a variety of ways, such as the bandwidth capability of each connection. Thus for example, connections 106 and 107 can have a relatively high bandwidth capability, while connection 108 can have a somewhat lower bandwidth capability and connections 109 and 110 can have a still lower bandwidth capability. Accordingly, cluster manager 118 can be configured to group destination devices 111, or communication links 106-110, into performance clusters, e.g., performance clusters 119, 120, and 121, based on their similar performance capabilities. In Figure 1, performance cluster 119 is shown as a high rate or high performance cluster; performance cluster 120 as shown as a medium rate or intermediate performance cluster; and performance cluster 121 is shown as a low rate or low performance cluster. The number of clusters needed may vary as system requirements dictate. The cluster manager can be further configured to assign a synchronization mechanism 103, 104, or 105 to each of the performance clusters 119, 120, and 121. Synchronization mechanisms 103, 104, or 105 can be configured to then send updates to associated destination devices 111 or associated communication links 106-110 in a manner that is optimized for the performance capabilities of the associated destination devices or communication links 106-110.

[020] Source device 101 can be configured to share data stored in the source data buffer 102 with the plurality of destination devices 111. Communication connections 106-110 provide mechanisms for transferring data, i.e., physical communications channels, while

the synchronization mechanisms 103-105 and corresponding synchronization mechanisms 116 associated with destination devices 111 can be configured to provide the computation and protocols needed to share data between source data buffer 102 and destination data buffers 117 over communication connections 106-110.

[021] Figure 2 is a diagram illustrating an example method for generating performance clusters, such as performance clusters 119, 120, and 121, in accordance with one embodiment of the systems and methods described herein. In step 302, the number of communication connections can be determined, e.g., by cluster manager 118. Cluster manager 118 can be configured to then determine, in step 304, the similarity in the performance capabilities of the communication connections. Based on the similarity determined in step 304, cluster manager can be configured to determine a required number of performance clusters in step 306. Cluster manager 118 can be configured to then cause the required number of synchronization mechanisms to be generated, in step 308 and then assign each communication connection to the appropriate performance cluster in step 310.

[022] In one embodiment, the similarity in performance capability of the various communication connections is determined, in step 304, by maintaining statistics for data rate capability of each communication connection. Such statistics can be derived, for example, from observing data transfers between source device 101 and destination devices 111. In another embodiment, connection security associated with each of the communication connections 106-110 can be used to determine similar performance capabilities in step 304. In still another embodiment, the error rate associated with data transfer of each communication connection can be used in step 304. In yet another embodiment, latency associated with data transfer of each communication connection can be used in step 304. In fact, it will be

understood that a variety of performance parameters and related information can be used to determine the similarity in the performance capabilities of the various communication connections. Thus, nothing within the specification or the claims that follow should be seen as limiting the systems and methods described herein to the use of any particular parameters or set of parameters.

[023] In one embodiment, the number of synchronization mechanisms can be determined dynamically and can change as needed to accommodate destination devices 111 as they connect with source device 101. Several algorithms can be used in selecting the number of synchronization mechanisms, some of which are described below. This dynamic capability can allow for a trade off between improved client service, which results when there are fewer destination devices 111 per cluster, and reduced server resource usage, which results from having a large amount of clusters. Thus, for example, if there are only a few destination devices 111, or if client service is important, then cluster manager can assign, for example, each destination device 111 to its own synchronization mechanism. On the other hand, if there are a lot of destination devices 111, or if client service is not as important, then cluster manager can assign fewer synchronization mechanisms. Moreover, depending on the embodiment, cluster manager 118 can be configured to dynamically update the destination device groupings and add or remove synchronization mechanisms as required.

[024] Further, in one particular embodiment, the correspondence between a destination device 111 and a particular synchronization mechanism 103, 104, or 105 can also be dynamic. In other words, the corresponding communication connection for a particular destination device 111 can be moved to a different synchronization mechanism if the corresponding performance capabilities change such that a different performance cluster 119,

120, or 121 is more appropriate. Thus, for example, cluster manager 118 can be configured to monitor, in step 312, a set of statistics associated with the performance of each communication connection 106-110 and to detect any change therein. If a significant change is detected, then the statistics can be used to determine if another performance cluster 119, 120, or 121 is more appropriate for the particular destination device 111.

[025] In one embodiment, all connections 106-110 or all destination devices 111 begin a session as part of a primary performance cluster. After a small number of updates the average latency for each destination device 111 or communication connection 106-110 is gathered. Cluster manager 118 can be configured to then perform a cluster division (step 306) to organize destination devices 111 according to their performance levels, e.g., their average latencies. For example, an initial calculation of the average latencies for each of the plurality of connections can be performed and used to determine the mean latency for the primary performance cluster. A standard deviation relative to the mean can also be calculated. The number of performance clusters required can then be determined based on the percentage of communication connections 106-110 with latencies within a certain number of standard deviations from the mean.

[026] In one particular embodiment, a minimum standard deviation threshold can be used in order to prevent the creation of extra clusters when the performance level is very similar.

[027] Further, in one embodiment, communication connections 106-110 can be placed into an appropriate performance cluster (step 308) using an algorithm, such as the K-means algorithm. The K-means algorithm is a partitioning method based on minimizing the sum-of-squares distance between average latencies for each communication connection 106-

110 and the mean latency for the primary performance cluster, allowing division of communication connections 106-110 into (*K*) performance clusters. This is an iterative approach that maximizes the ratio of variation between performance clusters relative to the variation within a performance cluster. This approach allows a quick calculation with a resulting distinct separation of performance levels. Depending on the embodiment, the K-means algorithm is executed periodically or as needed, e.g., if there is a change in performance capabilities (step 312) or a new destination device 111 joins the session (step 314).

[028] In step 314, the cluster manager 118 can monitor the communication connections 106-110 to detect new connection, i.e., a communication connection that is established after a session has begun. The cluster manager 118 can determine the performance capabilities of the new connection and add the new connection to a performance cluster based upon the performance capabilities of the new connection. In one embodiment, such communication connections will be inserted into a performance cluster (step 308) without moving other communication connections at least until some initial lag statistics are computed for the new communication connection. The new communication connection can, for example, then be moved into a performance cluster based on, e.g., a least squares analysis.

[029] A new insertion into a performance cluster can require a resynchronization for destination devices 111 within the performance cluster on the next cluster-wide update. In one embodiment, only the newly added destination device 111 need be resynchronized while the other destination devices 111 remain synchronized.

[030] Figure 3 is a flow chart illustrating an exemplary method for updating destination data buffers 117 associated with a particular performance cluster 119-121 in

accordance with one embodiment of the systems and methods described herein. Thus, in step 402, an update can be sent using a synchronization mechanism 103, 104, or 105. When the update is sent in step 402, a timer can then be started, in step 404, In step 406, the associated communication connections are monitored until one of the associated destination devices, requests another update. When another request is received the timer is stopped in step 408. The timer value can then be used to determine the latency associated with the communication connection for the requesting destination device 111. For each destination device 111 two metrics can be determined and maintained in step 410. The two metrics can include a "Total Session Average Latency" and a "Recent Average Latency." The "Total Session Latency" can be an average of all latency values associated with the requesting destination device 111. The "Recent Average Latency" can be an average of some number of the most recent latency values for the requesting destination device 111.

[031] A destination device 111 that does not respond within a timeout threshold, as determined in step 412, can be removed from its performance cluster, in step 414, so that other destination devices 111 in the performance cluster can still receive updates. This can, for example, prevent a network interruption or an issue associated with a destination device 111 from harming the other cluster participants' experiences. If a destination device 111 has reached this timeout, in step 412, but eventually responds, in step 416, then it can still be allowed to receive full-buffer updates, in step 418, e.g., until its Recent Average Latency performance merits insertion back into one of the performance clusters 119-121.

[032] In one embodiment, the synchronization mechanisms 103-105 and the synchronization mechanisms 116 can be configured to operate by dividing the data in source data buffer 102 into a number of blocks or sections. Initially, or whenever synchronization is

lost, a complete set of all blocks can be sent from source device 101 to a destination device 111. The associated destination device buffer 117 can then be updated using the complete set of blocks so that it is a replica of the source data buffer 102. Subsequently source device 101 can send only blocks that have changed subsequent to the last update sent to the destination data buffer 117. This approach can, for example, result in considerable savings in network bandwidth.

[033] In one embodiment, updates are sent when all connected devices in a cluster have requested an update.

[034] It should be noted that the original source data buffer can be located on a remote source device, e.g., a device that is not immediately connected with destination devices 111. For example, figure 4 is a diagram illustrating an example network communication system 200, which comprises a remote source device 205, with remote source data buffer 207, in accordance with one embodiment of the systems and methods described herein. Remote source device 205 can be interfaced with destination devices 111 via an intermediate source device 201. Thus, intermediate source device 201 can comprise a cluster manager 118, which can be configured to group communication connections 106-110 into performance clusters 119-121 using synchronization mechanisms 103-105.

[035] In addition, intermediate source device 201 can comprise an intermediate source data buffer 202, which can be kept in synchronization with remote source data buffer 207 using synchronization mechanisms 203 and 205. Destination data buffers 117 can then be kept in synchronization with intermediate source data buffer 202 as described above in relation to source data buffer 102.

[036] In one example of figure4, synchronization mechanism 103 can be configured to provide updates to performance cluster119, which comprises the highest data rate destination devices 111 and communication connections 106 and 107. The update interval required by synchronization mechanism 103 can thus determine the highest update rate needed and can therefore also serve as the update interval used by synchronization mechanisms 203 and 205 for updating intermediate source data buffer 202. This approach can be used to avoid redundant or excessive data requests for communication connection 204.

[037] While certain embodiments of the inventions have been described above, it will be understood that the embodiments described are by way of example only. Accordingly, the inventions should not be limited based on the described embodiments. Rather, the scope of the inventions described herein should only be limited in light of the claims that follow when taken in conjunction with the above description and accompanying drawings.